Pipeline para procesamiento con Freesurfer en cluster

Llamada a Freesurfer

Queremos hacer una llamada a Freesurfer para realizar un `-autorecon-all` () usando los volúmenes NIFTI de T1 y T2 (si este último existen) de una sesión de un sujeto. Esto se hace invocando:

```
$ recon-all -autorecon-all -i imgT1.nii -T2 imgT2.nii -subjid SUBJ_ID
```

Implementación del pipeline

Se implementa un pipeline que lleve a cabo los siguientes pasos:

1. descargar los NII del repositorio Xnat: se asume que estos archivos han sido creados en un proceso anterior con el pipeline DicomToNifti_Y
2. si éstos no existen en el repositorio Xnat, generarlos a partir de las series DICOM correspondientes, es decir
    1. descargar las series DICOM del repositorio Xnat
    2. ejecutar la conversión dcm2niix tal como se ejecuta desde DicomToNifti_Y
3. detectar cuáles son los volúmenes NIFTI correspondientes a T1 y T2 (si existe) y llamar a `recon-all`
4. comprimir en un archivo tar.gz los resultados
5. subirlos al repositorio Xnat como recursos del “experimento”

Nota: la estructura del pipeline es similar a los otros descritos en Plataforma XNAT - Pipelines, la única novedad son los *resources* añadidos para las instrucciones de línea de comandos `tar.xml` y `rmdir.xml`, y el uso de los atributos `precondition` y `continueOnFailure` para asegurar que los pasos de procesamiento se ejecutan cuando existen los archivos necesarios o que no detienen el pipeline en caso de error (respectivamente).

La llamada al programa `recon-all` se lleva a cabo en el *step* RUNFREESURFER

```
    <step id="RUNFREESURFER" description="Run freesurfer"
workdirectory="^/Pipeline/parameters/parameter[name='workdir']/values/unique
/text()^" continueOnFailure="false" >
        <!-- mkdir -p $WORKDIR/FS -->
        <resource name="mkdir" location="commandlineTools" >
            <argument id="dirname">
<value>^/Pipeline/parameters/parameter[name='fsdir']/values/unique/text()^</
value>
            </argument>
        </resource>
```

```
        <resource name="runFreesurfer" location="RunFreesurfer/resources">
            <argument id="dirNII">
<value>^/Pipeline/parameters/parameter[name='niidir']/values/unique/text()^<
/value>
            </argument>
            <argument id="tagT1">
<value>^/Pipeline/parameters/parameter[name='dcmT1tag']/values/unique/text()
^</value>
            </argument>
            <argument id="tagT2">
<value>^/Pipeline/parameters/parameter[name='dcmT2tag']/values/unique/text()
^</value>
            </argument>
            <argument id="dirFS">
<value>^/Pipeline/parameters/parameter[name='fsdir']/values/unique/text()^</
value>
            </argument>
            <argument id="subjectID">
<value>^/Pipeline/parameters/parameter[name='subjectID']/values/unique/text(
)^</value>
            </argument>
        </resource>
    </step>
```

El código del pipeline RunFreesurfer.xml es así

RunFreesurfer.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Pipeline xmlns="http://nrg.wustl.edu/pipeline"
xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://nrg.wustl.edu/pipeline
..\schema\pipeline.xsd"
xmlns:fileUtils="http://www.xnat.org/java/org.nrg.imagingtools.utils.Fi
leUtils">

<name>RunFreesurfer</name>

<location>RunFreesurfer</location>

<description>Launch the Freesurfer analysis pipeline</description>

<!-- to be interpreted by the schedule script -->
<resourceRequirements>
   <property name="srun_args">-p fast --cpus-per-task=4 --mem-per-
cpu=4G</property>
</resourceRequirements>
```

```
<!-- Standard input parameters -->
<documentation>
    <authors>
        <author>
            <lastname>Rodriguez-Perez</lastname>
            <firstname>Daniel</firstname>
        </author>
    </authors>
    <version>0</version>
    <input-parameters>
        <parameter>
            <name>scanIDs</name>
<values><schemalink>xnat:mrSessionData/scans/scan/ID</schemalink></valu
es>
            <description>The scan ids of all the scans of the
session</description>
        </parameter>
        <parameter>
            <name>sessionID</name>
<values><schemalink>xnat:mrSessionData/ID</schemalink></values>
            <description>Xnat session ID</description>
        </parameter>
        <parameter>
            <name>sessionName</name>
<values><schemalink>xnat:mrSessionData/label</schemalink></values>
            <description>Xnat session label</description>
        </parameter>
        <parameter>
            <name>projectID</name>
<values><schemalink>xnat:mrSessionData/project</schemalink></values>
            <description>Project</description>
        </parameter>
        <parameter>
            <name>subjectID</name>
<values><schemalink>xnat:mrSessionData/subject_ID</schemalink></values>
            <description>Subject ID</description>
        </parameter>
        <parameter>
            <name>dcmT1tag</name>
            <values><csv>_tfl3d1_16ns</csv></values>
            <description>Tags to detect T1</description>
        </parameter>
        <parameter>
            <name>dcmT2tag</name>
            <values><csv>_spcir_284ns</csv></values>
            <description>Tags to detect T2</description>
        </parameter>
    </input-parameters>
</documentation>

<xnatInfo appliesTo="xnat:mrSessionData"/>
```

```
<outputFileNamePrefix>^concat(/Pipeline/parameters/parameter[name='work
dir']/values/unique/text(),'/LOG')^</outputFileNamePrefix>

<!-- Pipeline loop -->
<loop id="series"
xpath="^/Pipeline/parameters/parameter[name='scanIDs']/values/list^"/>

<!-- Stats results loop -->
<loop id="stats"
xpath="^/Pipeline/parameters/parameter[name='statFiles']/values/list^"/
>

<!-- Other computed parameters -->
<parameters>
    <parameter>
        <name>output_nifti_filename_format</name>
        <values>
            <unique>%n_%s_%d</unique>
        </values>
        <description>Scan ids of all the scans of the
session</description>
    </parameter>
    <parameter>
        <name>workdir</name>
        <values>
<unique>^concat(/Pipeline/parameters/parameter[name='builddir']/values/
unique/text(),'/',/Pipeline/parameters/parameter[name='sessionName']/va
lues/unique/text())^</unique>
        </values>
    </parameter>
    <parameter>
        <name>dcmdir</name>
        <values>
<unique>^concat(/Pipeline/parameters/parameter[name='builddir']/values/
unique/text(),'/',/Pipeline/parameters/parameter[name='sessionName']/va
lues/unique/text(),'/DCM/')^</unique>
        </values>
    </parameter>
    <parameter>
        <name>niidir</name>
        <values>
<unique>^concat(/Pipeline/parameters/parameter[name='builddir']/values/
unique/text(),'/',/Pipeline/parameters/parameter[name='sessionName']/va
lues/unique/text(),'/NII/')^</unique>
        </values>
    </parameter>
    <parameter>
        <name>fsdir</name>
        <values>
```

```
<unique>^concat(/Pipeline/parameters/parameter[name='builddir']/values/
unique/text(),'/',/Pipeline/parameters/parameter[name='sessionName']/va
lues/unique/text(),'/FS/')^</unique>
        </values>
    </parameter>
    <parameter>
        <name>statFiles</name>
        <values>
            <list>aseg.stats</list>
            <list>lh.BA_exvivo.stats</list>
            <list>lh.BA_exvivo.thresh.stats</list>
            <list>lh.aparc.DKTatlas.stats</list>
            <list>lh.aparc.a2009s.stats</list>
            <list>lh.aparc.pial.stats</list>
            <list>lh.aparc.stats</list>
            <list>lh.curv.stats</list>
            <list>lh.w-g.pct.stats</list>
            <list>rh.BA_exvivo.stats</list>
            <list>rh.BA_exvivo.thresh.stats</list>
            <list>rh.aparc.DKTatlas.stats</list>
            <list>rh.aparc.a2009s.stats</list>
            <list>rh.aparc.pial.stats</list>
            <list>rh.aparc.stats</list>
            <list>rh.curv.stats</list>
            <list>rh.w-g.pct.stats</list>
            <list>wmparc.stats</list>
        </values>
    </parameter>
</parameters>


<steps>

    <step id="MKDIRS" description="Make NII and DCM folders"
workdirectory="^/Pipeline/parameters/parameter[name='workdir']/values/u
nique/text()^">
        <resource name="mkdir" location="commandlineTools" >
            <argument id="dirname">
<value>^/Pipeline/parameters/parameter[name='niidir']/values/unique/tex
t()^</value>
            </argument>
        </resource>
        <resource name="mkdir" location="commandlineTools" >
            <argument id="dirname">
<value>^/Pipeline/parameters/parameter[name='dcmdir']/values/unique/tex
t()^</value>
            </argument>
        </resource>
    </step>

    <step id="MKNIFTIDIR1" description="Make NIFTI folders"
```

```
workdirectory="^/Pipeline/parameters/parameter[name='niidir']/values/unique/text()^">
        <!-- mkdir $SERIES -->
        <resource name="mkdir" location="commandlineTools" >
            <argument id="dirname">
                <value>^PIPELINE_LOOPON(series)^</value>
            </argument>
        </resource>
    </step>
    <step id="DOWNLOADNIFTI" description="Download scan NIFTIs"
workdirectory="^concat(/Pipeline/parameters/parameter[name='niidir']/values/unique/text(),PIPELINE_LOOPON(series))^" continueOnFailure="true">
        <!-- XnatDataClient -username $USER -password $PASSWORD -useAbsolutePAth -batch -method GET "$HOST/data/experiments/$SESSIONID/scans/$SERIES/resources/NIFTI/files" -->
        <resource name="XnatDataClient" location="xnat_tools">
            <argument id="sessionId">
                <value>^fileUtils:getJSESSION('DUMMY')^</value>
            </argument>
            <argument id="absolutePath"/>
            <argument id="batch"/>
            <argument id="method">
                <value>GET</value>
            </argument>
            <argument id="remote">
<value>^concat(/Pipeline/parameters/parameter[name='host']/values/unique/text(),'/data/experiments/',/Pipeline/parameters/parameter[name='sessionID']/values/unique/text(),'/scans/',PIPELINE_LOOPON(series),'/resources/NIFTI/files')^</value>
            </argument>
        </resource>
    </step>
    <step id="RMEMPTYNIFTIDIR1" description="Remove empty NIFTI folders"
workdirectory="^/Pipeline/parameters/parameter[name='niidir']/values/unique/text()^" continueOnFailure="true">
        <!-- rmdir $SERIES -->
        <resource name="rmdir" location="commandlineTools" >
            <!-- resource not in standard xnat pipelines catalog -->
            <argument id="dir">
                <value>^PIPELINE_LOOPON(series)^</value>
            </argument>
        </resource>
    </step>

    <step id="MKDCMDIRS" description="Create directories to download DICOM"
precondition="!EXISTS(^concat(/Pipeline/parameters/parameter[name='niidir']/values/unique/text(),PIPELINE_LOOPON(series))^)"
```

```
workdirectory="^/Pipeline/parameters/parameter[name='dcmdir']/values/un
ique/text()^" continueOnFailure="true">
        <!-- mkdir $series -->
        <resource name="mkdir" location="commandlineTools">
            <argument id="dirname">
                <value>^PIPELINE_LOOPON(series)^</value>
            </argument>
        </resource>
    </step>
    <step id="DOWNLOADDCM" description="Downloads DCM"
precondition="EXISTS(^concat(/Pipeline/parameters/parameter[name='dcmdi
r']/values/unique/text(),PIPELINE_LOOPON(series))^)"
workdirectory="^concat(/Pipeline/parameters/parameter[name='dcmdir']/va
lues/unique/text(),PIPELINE_LOOPON(series))^" continueOnFailure="true">
        <!-- XnatDataClient -username $USER -password $PASSWORD -
useAbsolutePath -batch -method GET
"$HOST/data/experiments/$SESSIONID/scans/$SERIES/resources/NIFTI/files"
-->
        <resource name="XnatDataClient" location="xnat_tools">
            <argument id="sessionId">
                <value>^fileUtils:getJSESSION('DUMMY')^</value>
            </argument>
            <argument id="absolutePath"/>
            <argument id="batch"/>
            <argument id="method">
                <value>GET</value>
            </argument>
            <argument id="remote">
<value>^concat('"',/Pipeline/parameters/parameter[name='host']/values/u
nique/text(),'/data/experiments/',/Pipeline/parameters/parameter[name='
sessionID']/values/unique/text(),'/scans/',PIPELINE_LOOPON(series),'/re
sources/DICOM/files"')^</value>
            </argument>
        </resource>
    </step>
    <step id="RMEMPTYDCMDIR" description="Remove empty DCM folders"
workdirectory="^/Pipeline/parameters/parameter[name='dcmdir']/values/un
ique/text()^" continueOnFailure="true">
        <!-- rmdir $SERIES -->
        <resource name="rmdir" location="commandlineTools" >
            <!-- resource not in standard xnat pipelines catalog -->
            <argument id="dir">
                <value>^PIPELINE_LOOPON(series)^</value>
            </argument>
        </resource>
    </step>
    <step id="MKNIFTIDIR2" description="Make NIFTI folders"
precondition="EXISTS(^concat(/Pipeline/parameters/parameter[name='dcmdi
r']/values/unique/text(),PIPELINE_LOOPON(series))^)"
workdirectory="^/Pipeline/parameters/parameter[name='niidir']/values/un
ique/text()^">
```

```
        <!-- mkdir $SERIES -->
        <resource name="mkdir" location="commandlineTools" >
            <argument id="dirname">
                <value>^PIPELINE_LOOPON(series)^</value>
            </argument>
        </resource>
    </step>
    <step id="RUNNIFTY" description="Run dcm2nifti on DCM Dicoms if NII
files did not exist yet as resources"
precondition="EXISTS(^concat(/Pipeline/parameters/parameter[name='dcmdi
r']/values/unique/text(),PIPELINE_LOOPON(series))^)"
workdirectory="^/Pipeline/parameters/parameter[name='niidir']/values/un
ique/text()^" continueOnFailure="true">
        <resource name="dcm2niix" location="dcm2niix/resources">
            <argument id="gzip">
                <value>y</value>
            </argument>
            <argument id="ignore_derived"><!-- new argument -->
                <value>y</value>
            </argument>
            <argument id="output_filename_format">
<value>^/Pipeline/parameters/parameter[name='output_nifti_filename_form
at']/values/unique/text()^</value>
            </argument>
            <argument id="output">
<value>^concat(/Pipeline/parameters/parameter[name='niidir']/values/uni
que/text(),PIPELINE_LOOPON(series))^</value>
            </argument>
            <argument id="input">
<value>^concat(/Pipeline/parameters/parameter[name='dcmdir']/values/uni
que/text(),PIPELINE_LOOPON(series))^</value>
            </argument>
        </resource>
    </step>
    <step id="RMEMPTYNIFTIDIR2" description="Remove empty NIFTI
folders"
workdirectory="^/Pipeline/parameters/parameter[name='niidir']/values/un
ique/text()^" continueOnFailure="true">
        <!-- rmdir NII/$SERIES -->
        <resource name="rmdir" location="commandlineTools" >
            <!-- resource not in standard xnat pipelines catalog -->
            <argument id="dir">
                <value>^PIPELINE_LOOPON(series)^</value>
            </argument>
        </resource>
    </step>

    <step id="RUNFREESURFER" description="Run freesurfer"
workdirectory="^/Pipeline/parameters/parameter[name='workdir']/values/u
nique/text()^" continueOnFailure="false" >
```

```
        <!-- mkdir -p $WORKDIR/FS -->
        <resource name="mkdir" location="commandlineTools" >
            <argument id="dirname">
<value>^/Pipeline/parameters/parameter[name='fsdir']/values/unique/text
()^</value>
            </argument>
        </resource>

        <resource name="runFreesurfer"
location="RunFreesurfer/resources">
            <argument id="dirNII">
<value>^/Pipeline/parameters/parameter[name='niidir']/values/unique/tex
t()^</value>
            </argument>
            <argument id="tagT1">
<value>^/Pipeline/parameters/parameter[name='dcmT1tag']/values/unique/t
ext()^</value>
            </argument>
            <argument id="tagT2">
<value>^/Pipeline/parameters/parameter[name='dcmT2tag']/values/unique/t
ext()^</value>
            </argument>
            <argument id="dirFS">
<value>^/Pipeline/parameters/parameter[name='fsdir']/values/unique/text
()^</value>
            </argument>
            <argument id="subjectID">
<value>^/Pipeline/parameters/parameter[name='subjectID']/values/unique/
text()^</value>
            </argument>
        </resource>
    </step>

    <step id="ZIPRESULTS" description="Zip results"
workdirectory="^/Pipeline/parameters/parameter[name='fsdir']/values/uni
que/text()^" continueOnFailure="false" >
        <!-- tar -zcf $SUBJECTID.tar.gz * -->
        <resource name="tar" location="commandlineTools" >
            <!-- resource not in standard xnat pipelines catalog -->
            <argument id="compress"/>
            <argument id="archive">
<value>^concat(/Pipeline/parameters/parameter[name='sessionName']/value
s/unique/text(),'.tar.gz')^</value>
            </argument>
            <argument id="files">
                <!--
value>^/Pipeline/parameters/parameter[name='subjectID']/values/unique/t
ext()^< / value -->
                <value>*</value>
            </argument>
        </resource>
```

```
    </step>

    <step id="UPLOADRESULTS" description="Upload results as resources"
workdirectory="^/Pipeline/parameters/parameter[name='fsdir']/values/uni
que/text()^" continueOnFailure="false" >
        <!-- XnatDataClient -username $USER -password $PASSWORD -
useAbsolutePAth -batch -method PUT
"$HOST/data/experiments/$SESSIONID/assets" -->
        <resource name="XnatDataClient" location="xnat_tools">
            <!-- do not use sessionId because it becomes outdated all
too soon -->
            <!-- argument id="sessionId">
             <value>^fileUtils:getJSESSION('DUMMY')^</value>
         </argument -->
            <argument id="user">
<value>^/Pipeline/parameters/parameter[name='user']/values/unique/text(
)^</value>
            </argument>
            <argument id="password">
<value>^/Pipeline/parameters/parameter[name='pwd']/values/unique/text()
^</value>
            </argument>
            <argument id="method">
                <value>PUT</value>
            </argument>
            <argument id="remote">
<value>^concat('"',/Pipeline/parameters/parameter[name='host']/values/u
nique/text(),'/data/experiments/',/Pipeline/parameters/parameter[name='
sessionID']/values/unique/text(),'/resources/FS/files?overwrite=true&am
p;format=TAR&amp;content=FSresults&amp;reference=',/Pipeline/parameters
/parameter[name='fsdir']/values/unique/text(),/Pipeline/parameters/para
meter[name='sessionName']/values/unique/text(),'.tar.gz&amp;event_id=',
/Pipeline/parameters/parameter[name='workflowid']/values/unique/text(),
'"')^</value>
            </argument>
        </resource>
    </step>

    <step id="UPLOADSTATS" description="Upload stats as resources"
workdirectory="^/Pipeline/parameters/parameter[name='fsdir']/values/uni
que/text()^" continueOnFailure="false" >
        <!-- XnatDataClient -username $USER -password $PASSWORD -
useAbsolutePAth -batch -method PUT
"$HOST/data/experiments/$SESSIONID/resources/FS" -->
        <!-- XnatDataClient -u $USER -p $PASSWORD  -m PUT -l
tmp/SequenceNames.txt
"$HOST/data/experiments/XNAT_E00090/resources/FS/files/myfile2.txt?over
write=true&format=TXT&content=FSresults&inbody=false&file=upload" -->
        <resource name="XnatDataClient" location="xnat_tools">
            <!-- do not use sessionId because it becomes outdated all
```

```
too soon -->
            <!-- argument id="sessionId">
             <value>^fileUtils:getJSESSION('DUMMY')^</value>
         </argument -->
            <argument id="user">
<value>^/Pipeline/parameters/parameter[name='user']/values/unique/text(
)^</value>
            </argument>
            <argument id="password">
<value>^/Pipeline/parameters/parameter[name='pwd']/values/unique/text()
^</value>
            </argument>
            <argument id="method">
                <value>PUT</value>
            </argument>
            <argument id="infile">
<value>^concat(/Pipeline/parameters/parameter[name='fsdir']/values/uniq
ue/text(),
/Pipeline/parameters/parameter[name='subjectID']/values/unique/text(),
'/stats/', PIPELINE_LOOPON(stats))^</value>
            </argument>
            <argument id="remote">
<value>^concat('"',/Pipeline/parameters/parameter[name='host']/values/u
nique/text(),'/data/experiments/',/Pipeline/parameters/parameter[name='
sessionID']/values/unique/text(),'/resources/FS/files/',
PIPELINE_LOOPON(stats),
'?overwrite=true&amp;format=TXT&amp;content=FSstats&amp;inbody=false&am
p;file=upload&amp;event_id=',/Pipeline/parameters/parameter[name='workf
lowid']/values/unique/text(),'"')^</value>
            </argument>
        </resource>
    </step>

    <!-- notify the user with an e-mail message -->
    <step id="ENDNOTIFY" description="Notify" continueOnFailure="true">
        <resource name="Notifier" location="notifications">
            <argument id="user">
<value>^/Pipeline/parameters/parameter[name='user']/values/unique/text(
)^</value>
            </argument>
            <argument id="password">
<value>^/Pipeline/parameters/parameter[name='pwd']/values/unique/text()
^</value>
            </argument>
            <argument id="cc">
<value>^/Pipeline/parameters/parameter[name='adminemail']/values/unique
/text()^</value>
            </argument>
            <argument id="from">
<value>^/Pipeline/parameters/parameter[name='adminemail']/values/unique
/text()^</value>
```

```
            </argument>
            <argument id="to">
<value>^/Pipeline/parameters/parameter[name='useremail']/values/unique/
text()^</value>
            </argument>
            <argument id="subject">
<value>^concat(/Pipeline/parameters/parameter[name='xnatserver']/values
/unique/text(), ' update: ',
/Pipeline/parameters/parameter[name='session']/values/unique/text(),'
FreeSurfer processing complete')^</value>
            </argument>
            <argument id="host">
<value>^/Pipeline/parameters/parameter[name='mailhost']/values/unique/t
ext()^</value>
            </argument>
            <argument id="body">
                <value>^concat('Dear ',
/Pipeline/parameters/parameter[name='userfullname']/values/unique/text(
), ',&lt;br&gt; &lt;p&gt; The FreeSurfer pipeline completed without
errors.&lt;/p&gt; &lt;br&gt; &lt;p&gt; Details for this analysis are
available at &lt;a href="',
/Pipeline/parameters/parameter[name='host']/values/unique/text(),
'/app/action/DisplayItemAction/search_element/xnat:mrSessionData/search
_field/xnat:mrSessionData.ID/search_value/',
/Pipeline/parameters/parameter[name='sessionID']/values/unique/text(),
'/popup/false/',
/Pipeline/parameters/parameter[name='projectID']/values/unique/text(),
'"&gt; the ',
/Pipeline/parameters/parameter[name='xnatserver']/values/unique/text(),
' website.&lt;/a&gt; &lt;/p&gt; &lt;br&gt;',
/Pipeline/parameters/parameter[name='xnatserver']/values/unique/text(),
' Team.')^</value>
            </argument>
        </resource>
    </step>

</steps>
</Pipeline>
```

El *resource* `runFreesurfer.xml` describe el comportamiento del script Bash `runFreesurfer.sh` que selecciona los archivos NII a partir del tipo de secuencia que se puede leer en los JSON asociados a ellos. Ese identificador de secuencia (uno para las secuencias T1 y otro para las T2) se le pasan al *pipeline* como parámetros desde Xnat (los *parameters* `dcmT1tag` y `dcmT2tag`).

El código de runFreesurfer.xml es así

runFreesurfer.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Parse options and run recon-all -->
<Resource xmlns="http://nrg.wustl.edu/pipeline">
    <name>runFreesurfer.sh</name>
    <location>RunFreesurfer/scripts</location>
    <commandPrefix>bash</commandPrefix>
    <type>Executable</type>
    <description>Finds T1 and T2 and runs freesurfer script for the
subject</description>
    <estimated_time>05:00:00</estimated_time>
    <input>
        <argument id="dirNII" prefix="--">
            <name>dirNII</name>
            <description>NII input directory</description>
        </argument>
        <argument id="tagT1" prefix="--">
            <name>tagT1</name>
            <description>Tag to select T1</description>
        </argument>
        <argument id="tagT2" prefix="--">
            <name>tagT2</name>
            <description>Tag to select T2</description>
        </argument>
        <argument id="dirFS" prefix="--">
            <name>dirFS</name>
            <description>FS output directory</description>
        </argument>
        <argument id="subjectID" prefix="--">
            <name>subjectID</name>
            <description>Subject ID</description>
        </argument>
    </input>
</Resource>
```

El código del script Bash runFreesurfer.sh es así

runFreesurfer.sh

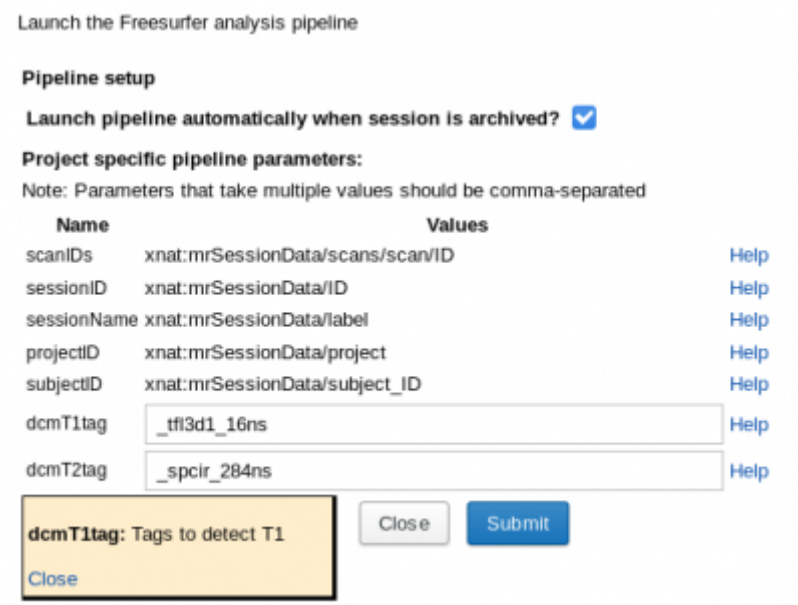
```
#!/bin/bash

ARGS=( "$@" )

#parse options
for ((n=0; n<${#ARGS[@]}; n++)) ; do
    case "${ARGS[$n]}" in
    --dirNII)
        let n=n+1
        dirNII="${ARGS[$n]}"
```

```
        #echo "dirNII=$dirNII"
        ;;
    --tagT1)
        let n=n+1
        tagT1="${ARGS[$n]}"
        #echo "tagT1=$tagT1"
        ;;
    --tagT2)
        let n=n+1
        tagT2="${ARGS[$n]}"
        #echo "tagT2=$tagT2"
        ;;
    --dirFS)
        let n=n+1
        dirFS="${ARGS[$n]}"
        #echo "dirFS=$dirFS"
        ;;
    --subjectID)
        let n=n+1
        subjectID="${ARGS[$n]}"
        #echo "subjectID=$subjectID"
        ;;
    esac
done

#check if all required options are present
if [ -z "$dirNII" ] || [ -z "$tagT1" ] || [ -z "$dirFS" ] || [ -z
"$subjectID" ] ; then
    echo "Error: not enough arguments" >&2
    exit 1
fi

#we are in the workdir
WDIR=$PWD

#parse JSON for T1
if ! [ -z "$tagT1" ] ; then #always true
    cd "$dirNII"
    for nij in */*.json ; do
        if grep SequenceName "$nij" | grep -q "$tagT1" ; then
            niiT1="$dirNII/$(echo "$nij" | sed 's/\.json$/\.nii.gz/')"
        fi
    done
    cd "$WDIR"
fi
#parse JSON for T2
if ! [ -z "$tagT2" ] ; then
    cd "$dirNII"
    for nij in */*.json ; do
        if grep SequenceName "$nij" | grep -q "$tagT2" ; then
```

```
            niiT2="$dirNII/$(echo "$nij" | sed 's/\.json$/\.nii.gz/')"
        fi
    done
    cd "$WDIR"
fi

#check if all required options are present
if [ -z "$niiT1" ] ; then
    echo "Error: no T1 sequence available" >&2
    exit 1
fi

#set up the environment
#export SUBJECTS_DIR=/nas/data/subjects
export SUBJECTS_DIR="$dirFS"
export FREESURFER_HOME=/usr/local/freesurfer
source $FREESURFER_HOME/SetUpFreeSurfer.sh
export PATH=$PATH:$FREESURFER_HOME/bin

#create subject directories
cd "$dirFS"
#mksubjdirs "$subjectID"

#run recon-all with one or two images
if [ -z "$niiT2" ] ; then
    recon-all -autorecon-all -i "$niiT1" -subjid "$subjectID"
else
    recon-all -autorecon-all -i "$niiT1" -T2 "$niiT2" -subjid
"$subjectID"
fi
```

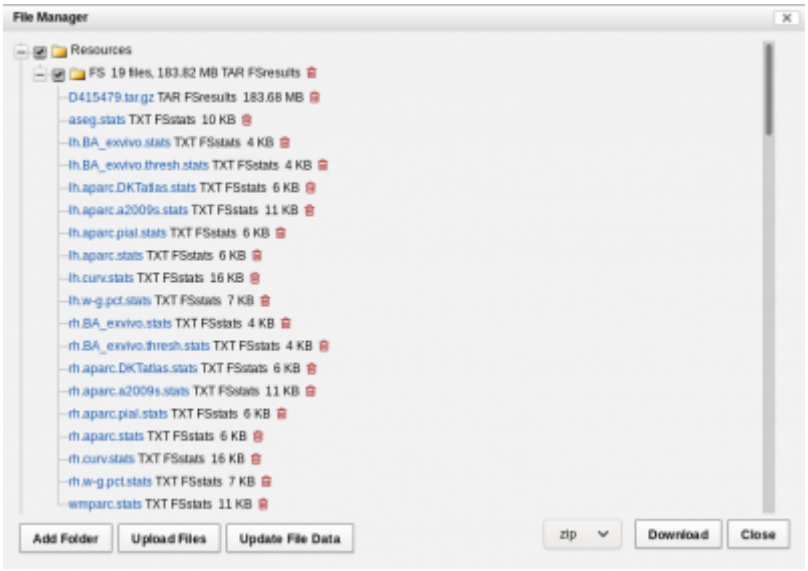
Ejecución del pipeline desde la GUI de Xnat

El *pipeline* se puede añadir a un proyecto de Xnat usando la interfaz gráfica.



Desde el diálogo de configuración se puede seleccionar que se ejecute al cargar un nuevo estudio. También se pueden indicar los nombres de las secuencias que se reconocerán como T1 y T2.

Una vez se ha ejecutado el *pipeline* los resultados se pueden encontrar en los “resources” del experimento, bajo la “carpeta” FS.



Ejecución del pipeline en el cluster usando Slurm

El mecanismo de pipelines de Xnat contempla la posibilidad de utilizar un *scheduler* para ejecutar los pipelines aprovechando una arquitectura cluster (o grid). No obstante, aunque la especificación existe, no existe una implementación (al menos, no una funcional para Slurm).

La especificación en los *pipelines* de Xnat consta de dos partes: los *resourceRequirements* del *pipeline* y el script `schedule` mediante el que se ejecuta el comando `PipelineRunner` que interpreta y lleva a cabo los pasos del procesamiento.

Los resourceRequirements del pipeline

Se trata de un elemento XML que se inserta antes de la `<documentation>` del *pipeline* y que define una serie de propiedades (pares clave-valor) que debería interpretar el *scheduler* en el que se ejecutase el *pipeline*; por esta razón, estas propiedades y sus valores son específicos de dicho *scheduler*.

Éste sería un ejemplo del código que usaremos para ejecutarlo en `slurm`:

```
<resourceRequirements><!-- to be interpreted by the schedule script -->
   <property name="srun_args">-p fast --cpus-per-task=4 --mem-per-
cpu=4G</property>
</resourceRequirements>
```

El script schedule

El script `schedule` (que se halla en `$PIPELINE_HOME/bin`) contiene solamente

```
#!/bin/bash
$@
```

Es decir, se limita a ejecutar los argumentos que se le pasan.

Un `schedule` más inteligente, que identifica el archivo XML del *pipeline* que es pasado mediante la opción `-pipeline` al programa `PipelineRunner`, lee en los `resourceRequirements` el `srun_args` y ejecuta el comando `srun` de acuerdo a ellos se incluye en el siguiente código:

schedule

```
#!/bin/bash

ARGS=( "$@" )
for ((n=0; n<${#ARGS[@]}; n++)) ; do
    case "${ARGS[$n]}" in
    -pipeline)
        let n=n+1
        XMLfile="${ARGS[$n]}"
    ;;
    esac
done

if ! [ -z "$XMLfile" ] &&\
    SARGS=( $( grep '<property name="srun_args">' "$XMLfile" | sed
's/^.*<property name="srun_args">//' | sed 's/<\/property>//' ) ) ;
then
    srun ${SARGS[@]} $@
else
    $@
```

```
fi
```